

Diffusion-KLMS Algorithm and its Performance Analysis for Non-Linear Distributed Networks

Rangeet Mitra, and Vimal Bhatia, *Senior Member, IEEE*

Abstract

In a distributed network environment, the diffusion-least mean squares (LMS) algorithm gives faster convergence than the original LMS algorithm. It has also been observed that, the diffusion-LMS generally outperforms other distributed LMS algorithms like spatial LMS and incremental LMS. However, both the original LMS and diffusion-LMS are not applicable in non-linear environments where data may not be linearly separable. A variant of LMS called kernel-LMS (KLMS) has been proposed in the literature for such non-linearities. In this paper, we propose kernelised version of diffusion-LMS for non-linear distributed environments. Simulations show that the proposed approach has superior convergence as compared to algorithms of the same genre. We also introduce a technique to predict the transient and steady-state behaviour of the proposed algorithm. The techniques proposed in this work (or algorithms of same genre) can be easily extended to distributed parameter estimation applications like cooperative spectrum sensing and massive multiple input multiple output (MIMO) receiver design which are potential components for 5G communication systems.

Index Terms

KLMS, Algorithm, Diffusion-LMS, Distributed Adaptive Filtering, Massive MIMO, Cognitive Radio

I. INTRODUCTION

Nowadays, there is a thrust toward development of a new standard for communications called 5G, which involves some novel approaches like massive multiple input multiple output (MIMO), cooperative spectral sensing, visible light communication (VLC) etc. [1]. Massive MIMO uses a large number of antenna array elements (which consist of antennae at the receiver and those at the network nodes) which greatly increases the capacity of the communication system. Spectral sensing is a technique to estimate vacant spectral subbands adaptively. Such vacant subbands may be used to accommodate incoming transmission which saves bandwidth as we are saved from allocating a new frequency band for the incoming signal. The distributed diffusion based adaptive filtering algorithms have potential applications in cooperative spectral sensing and distributed MIMO detection [2], [3], [4]. Hence distributed adaptive filtering/optimization over distributed networks is an important and emerging research area which can be applied to 5G standard components.

Distributed signal processing deals with drawing inferences from data coming from various nodes in a given graph. Robust distributed algorithms are required to draw inferences from the intelligently fused data from all the nodes. The task of training an artificial computer to automatically draw inferences and take decisions is assigned to the statistical learning techniques.

This paper is a preprint of a paper submitted to IET Signal Processing (special issue on 5G wireless networks) and is subject to Institution of Engineering and Technology Copyright. If accepted, the copy of record will be available at IET Digital Library

Statistical learning algorithms may be categorised into four distinct classes: a) Supervised learning, b) Unsupervised learning, c) Semi-supervised learning and d) Reinforcement learning [5]. In supervised learning, the data labels are assumed to be known during training. In unsupervised learning, the data labels are not known while training. In semi-supervised learning, only a subset of the labels are known. In reinforcement learning, the algorithm is trained in such a way so as to maximise a utility function. The scope of this paper is limited to distributed supervised learning.

One of the well known supervised learning rules is the Widrow-Hoff learning rule or the least mean squares (LMS) algorithm. It belongs to the class of stochastic gradient algorithms. It replaces the expectation operator in the Weiner-Hopf equation [6] by the instantaneous gradient of the quadratic cost function. In the recent literature [7], [8], there has been a major thrust towards generalising the LMS algorithm in distributed environments. A variant of the widely known LMS algorithm or the Widrow-Hoff learning rule, called the diffusion-LMS, has been used in distributed optimisation in [7] with wide number of application areas. This algorithm uses stochastic matrices to fuse the data intelligently coming from different sources (for example, nodes of the network) and has the best performance among all distributed counterparts of LMS algorithm [9], [10]. Similarly other extensions of adaptive filtering algorithms like recursive least squares (RLS) called diffusion-RLS have also been proposed [11].

Classical adaptive filtering algorithms like LMS and diffusion-LMS (for networks) work well for affinely separable data. However, in scenarios when the data is not guaranteed to be affinely separable [5], which occurs frequently in non-linear scenario, the kernel least mean squares (KLMS) algorithm has been found in the literature to perform better as demonstrated in [12] and has found wide applicability as in [13], [14], [15]. The basic principle of KLMS is the kernel trick [5], which maps the input data into a linearly separable high dimensional reproducing kernel Hilbert space (RKHS) [12]. Similar extensions to linear algorithms like affine projection algorithm to kernel spaces exist as in [16]. Kernel based distributed learning algorithms have been proposed in the literature [17], [18], [19]. However, they neither address the kernel LMS regression problem in the diffusion framework nor is their performance analysed in terms of popular performance metrics.

In this paper, we propose an extension of KLMS for distributed networks. In other words, we seek to apply the kernel trick to the diffusion-LMS adaptations given in [10]. We also seek to provide theoretical expressions that govern the proposed algorithm's transient and steady state behaviour as has been done in [10], [20] by classical adaptive filtering theory based approaches as given in [21].

This paper is organised as follows: to facilitate understanding of background material and concepts forming theoretical basis of the proposed algorithm, the diffusion-LMS algorithm and KLMS algorithm are reviewed in Section-II and Section-III respectively. The diffusion KLMS algorithm is proposed in section-IV. To gain insights into the performance of the algorithm, transient performance, steady-state performance and condition for convergence are mathematically analysed in Section-V. The simulation results and comparison with other algorithms is provided in Section-VI, and Section-VII concludes the paper.

II. REVIEW OF DISTRIBUTED DIFFUSION LMS

In this section, we review the distributed diffusion-LMS given as given in [10]. In the distributed diffusion-LMS algorithm, there are a set of nodes in a graph \mathcal{G} . The neighbourhood of a node in a graph is given by a set of nodes \mathcal{G}' such that there

exists an edge between that node and the nodes in the set \mathcal{G}' . Please note that for each node, \mathcal{G}' also includes the node itself. Let stochastic matrices be given by the entries $A = [a_{ij}]$ and $C = [c_{ij}]$ represent a probabilistic weight from node i to node j . This matrix is generally determined by stochastic sampling techniques as given in [10].

For a distributed adaptive graph indexed by time variable n , the adaptive filter attempts to estimate the local cost $J_q(n)$ function at time instant n at a given node:

$$J_q(n) = \sum_{l \in \mathcal{G}'} c_{lq} J_l(n) \quad (1)$$

where l runs over all members of the neighbourhood of the q^{th} node of the network and forms the q^{th} local cost function J_q . For this, the distributed Weiner solution based local estimate at node q will be, w_q^o , and is given as:

$$w_q^o(n) = \left(\sum_{l \in \mathcal{G}'} c_{lq} R_{x_{l_n}} \right)^{-1} \left(\sum_{l \in \mathcal{G}'} c_{lq} r_{dx_{l_n}} \right) \quad (2)$$

where, $R_{x_{l_n}}$ is the autocorrelation matrix for the l^{th} node in the neighbourhood of the node q of the graph. $r_{dx_{l_n}}$ is the cross correlation between the desired output d and x_l is the data from l^{th} member of the neighbourhood of node q at time n .

The weight vector w_q , for the q^{th} node, is iteratively adapted by diffusion-LMS as follows,

$$p_q(n+1) = p_q(n) + \mu \sum_{l \in \mathcal{G}'} c_{lq} (d_l(n) - w_l(n)^T x_l) x_l \quad (3)$$

$$w_q(n+1) = \sum_{l \in \mathcal{G}'} a_{lq} p_l(n+1) \quad (4)$$

where, μ is the step-size, $d_l(n)$ is the desired response at the l^{th} node at the n^{th} time instant and $p_q(n)$ is the vector of intermediate value of the adaptive filter at q^{th} node at n^{th} time instant before it can be combined probabilistically over its neighbourhood to get the final updated estimate.

The steps in eq. (3) and (4) can be carried out in either order. In both situations, it will belong to the same genre of algorithms. If the eq. (3) is carried out first it is called Adapt and Then Combine (ATC) diffusion. If the eq. (4) is carried out first it is called Combine and Then Adapt (CTA) diffusion [10].

Please note that an important factor in convergence of the adaptive filters is the spectral radius of the covariance matrix. This spectral radius is a norm in itself. Applying Jensen's inequality to the spectral radius as in [10], ρ_{max} of the weighted covariance matrix,

$$\rho_{max} \left(\sum_{l \in \mathcal{G}'} c_{lq} R_{lq} \right) \leq \sum_{l \in \mathcal{G}'} c_{lq} \rho_{max}(R_{lq}) \leq \max_{1 \leq l \leq N} \rho(R_{lq}) \quad (5)$$

where, $N = |\mathcal{G}'|$ and R_{lq} is the autocorrelation matrix of the l^{th} neighbour of the q^{th} node. Hence, due to lower eigen-value spread, it converges faster. More rigorous convergence results are found in [10].

III. REVIEW OF KLMS

The linear-LMS as described in [5], [6] minimises the following cost function at n^{th} instant:

$$J_{LMS}(n) = \mathbb{E}[(d(n) - w(n)^T x_n)^2] \quad (6)$$

where x_n is the observation vector for the n^{th} time instant and $\mathbb{E}[\cdot]$ is the expectation operator. Dropping the expectation operator and taking gradient with respect to w , we arrive at the following stochastic gradient update rule [5],

$$w(n+1) = w(n) + \mu e_{LMS}(n)x_n \quad (7)$$

where, $e_{LMS}(n) = (d(n) - w(n)^T x_n)$

When the data is not linearly separable the above adaptation does not converge to optimum value. Hence, in such scenarios, we invoke the kernel trick and map the vectors to RKHS as in [5] by a feature map $\phi : \mathbb{R}^m \rightarrow \mathcal{H}$.

In RKHS, the adaptation can be written as follows:

$$\Omega(n) = \Omega(n-1) + \mu e_{KLMS}(n-1)\phi(x_{n-1}) \quad (8)$$

where Ω is the implicit parameter to be estimated in RKHS. This can be written as a running summation as follows:

$$\Omega(n) = \mu \sum_{i=0}^{n-1} e_{KLMS}(i)\phi(x_i) \quad (9)$$

Taking inner product with the latest observation and assumption of zero initial conditions would give the following recursion as in [12]:

$$y(n+1) = \mu \sum_{i=0}^{k-1} e_{KLMS}(i) \langle \phi(x_i), \phi(x_n) \rangle_{\mathcal{H}} \quad (10)$$

where,

$$e_{KLMS}(n) = (d(n) - y(n)) \quad (11)$$

is the error at n^{th} instant and $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes a real kernel inner product [12] on RKHS \mathcal{H} . Several possibilities of kernel inner products exist; some of them being polynomial and Gaussian kernels [5]. This algorithm has a nice self-regularising property, and has been studied in details in [12].

IV. PROPOSED DIFFUSION-KLMS

Based on the KLMS algorithm, reviewed in the previous section, we propose its distributed variant in this section based on the diffusion approach. We now define matrices and symbols that will be used in this paper. In this proposal, we have the matrix $Y = [y(l, n)]$ to denote output corresponding to the l^{th} neighbour at n^{th} time instant. $E = [e(l, n)]$ is the error matrix corresponding to the l^{th} neighbour at n^{th} time instant. $X = [\{x_l(n)\}]$ is a matrix of measurement vectors from neighbours of node q at time instant n stacked together. In the following few lines, we will denote the collection of the data from various nodes at the n^{th} time instant as $X(n)$. $X(n)$ contains the data pertaining to all l neighbours stacked in row vector form. In case, there is no vector from a node in the neighbourhood it is replaced by the zero vector in $X(n)$ and will have a corresponding 0 entry in C .

The gradient from eq. (3) is redefined as:

$$\nabla_{p_q} J_q(n) = e(l, n)' \phi(CX(n)) \quad (12)$$

where $\phi(\cdot)$ is a feature map from $\mathbb{R}^d \rightarrow \mathcal{H}$, where d is the dimensionality of the data and \mathcal{H} is an RKHS. Applying the kernel trick results in,

$$y(l, n+1) = \mu \sum_{i=0}^{n-1} e(l, n)' < CX(i), X(n) >_{\mathcal{H}} \quad (13)$$

$$e(q, n+1)' = \sum_{l \in \mathcal{G}'} a(q, l) d_l(n) - \sum_{l \in \mathcal{G}'} a(q, l) y(l, n) \quad (14)$$

where A is a stochastic matrix corresponding to the probabilistic weights $\{a(q, l)\}$. The error at n^{th} time instant at the q^{th} node would be the (transformed) mean (by A) of e over all possible l .

The proposed algorithm is given below, as iterating following three steps, till convergence:

- 1) Estimate the outputs of node l using estimates of error $e'_l(n)$.
- 2) Form an estimate of errors at time instant n at each node l . Let this be given by the vector $e(n)$ whose l^{th} element is $e(l, n)$. Then the error term for the l^{th} node for the n^{th} time instant can be written as $e(l, n) = d(n) - y(l, n)$
- 3) The error at each node is modified by the transformation A by the equation $e'(n+1) = Ae(n)$, where $e(n)$ and $e'(n)$ are vectors of error terms corresponding to all the nodes (for all nodes indexed by l) stacked together.

V. TRANSIENT AND STEADY STATE PERFORMANCE

In this section, we provide the steady state analysis of the proposed algorithm based on the classical approach outlined in [21] (analysis based on eigenvalues of autocorrelation matrices). We note that the proposed recursion for the q^{th} node can be expressed in RKHS as follows:

$$\Omega_q(n) = \Omega_q(n-1) - \sum_{\forall l} e_q(n) c_{lq} \phi(x_l) \quad (15)$$

$$y_q(n) = < \Omega_q(n), \phi(x_{obs}) >_{\mathcal{H}}$$

$$e_q(n) = d_q(n) - y_q(n)$$

$$e_q(n+1) = \sum_{\forall l} a_{lq} e_l(n)$$

Ω_q is an implicit parameter which is learned in RKHS and x_{obs} is an input observation. Let the optimal value of the parameter be Ω^o and the deviation of the implicit parameter from the optimal value in q^{th} node at n^{th} instant be denoted as $\tilde{\Omega}_q(n)$. Subtracting Ω^o from both sides of first equation of (15), we get:

$$\tilde{\Omega}_q(n) = \tilde{\Omega}_q(n-1) - \sum_{\forall l} e_q(n) c_{lq} \phi(x_l) \quad (16)$$

Taking inner product on both sides of the above equation with $\phi(x_l)$,

$$\begin{aligned}\tilde{y}_q(n) &= \tilde{y}_q(n-1) - \mu \sum_{\forall l} c_{lq} (\tilde{y}_q(n-1) + n_q) \langle \phi(x_l), \phi(x_{obs}) \rangle_{\mathcal{H}} \\ &= (1 - \mu \sum_{\forall l} c_{lq} \langle \phi(x_l), \phi(x_{obs}) \rangle_{\mathcal{H}}) \tilde{y}_q(n-1) - \mu \sum_l c_{lq} n_q \langle \phi(x_l), \phi(x_{obs}) \rangle_{\mathcal{H}}\end{aligned}\quad (17)$$

Please note that $\tilde{y}_q(n)$ is calculated after combination by the A matrix in the last step of eq. (15). Define a matrix $A_1 = A \otimes I_D$ and $C_1 = C \otimes I_D$, where A and C are combining matrices and I_D is a $D \times D$ identity matrix; where D is the cardinality of the network. Further we define two vectors namely, $\Phi(x) = [\phi(x_1), \phi(x_2), \dots, \phi(x_d)]^T$ and $\Phi(x_{obs}) = [\phi(x_{obs}), \phi(x_{obs}), \dots, \phi(x_{obs})]^T$. Using above defined variables, we rewrite (17) as,

$$\tilde{y}_q(n) = (1 - \mu \langle C_1 \Phi(x), \Phi(x_{obs}) \rangle_{\mathcal{H}}) \tilde{y}_q(n-1) - \mu \langle C_1 \Phi(x), \Phi(x_{obs}) \rangle_{\mathcal{H}} n_q \quad (18)$$

Squaring both sides, taking expectation, and considering only till the first power of μ , we get:

$$\mathbb{E}[|\tilde{y}_q(n)|^2] = [1 - 2\mu \langle C_1 \Phi(x), \Phi(x_{obs}) \rangle_{\mathcal{H}}] \mathbb{E}[|\tilde{y}_q(n-1)|^2] + \mu^2 \sigma_n^2 \mathbb{E}[\langle C_1 \Phi(x), \Phi(x_{obs}) \rangle_{\mathcal{H}}^2] \quad (19)$$

Based on (19) we derive the transient behaviour, steady state behaviour and condition for convergence of the proposed algorithm.

A. Transient behaviour

To estimate the speed of convergence of the proposed approach it is essential to gain insight into the dynamical equation that governs the evolution of the learning curve vs number of iterations.

The above dynamical equation (19) controls the transient behaviour at small step-sizes. The inner product $\langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$ depends on choice of kernel. As we use a real Gaussian kernel as done in [12],

$$\langle \phi(x), \phi(y) \rangle_{\mathcal{H}} = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right) \quad (20)$$

where $x, y \in \mathbb{R}^m$ and $\phi : x \rightarrow \phi(x)$ is a feature map from the vector space of real numbers to RKHS. Using the definition of $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ given in (20) in (19) we get the transient behaviour of the proposed approach. We see that for a given μ and noise variance σ_n^2 the transient behaviour of the proposed approach can be easily modeled using (19).

B. Steady state behaviour

It is also essential to see how the MSE floor to which the proposed algorithm has converged varies with step-size. From (19), assuming convergence ($\mathbb{E}[|\tilde{y}_q(n)|^2] \approx \mathbb{E}[|\tilde{y}_q(n-1)|^2]$), we arrive at the following expression for misadjustment,

$$\mathbb{E}[|\tilde{y}_q(n)|^2] = \frac{\mu \sigma_n^2}{2} \mathbb{E}[\langle C_1 \Phi(x), \Phi(x_{obs}) \rangle_{\mathcal{H}}] \quad (21)$$

Thus we can see that the above equation (derived for non-linear systems) is similar to equation derived in [21] for the Widrow-Hopf learning rule for single node for linear parameter estimation.

C. Step-size range for convergence

For any adaptive algorithm it is very important to set up the step-size, μ , in the range in which the algorithm converges. If μ is too less, we may observe slow convergence. Too high a μ may result in mis-convergence.

The proposed algorithm converges iff the following condition holds,

$$\begin{aligned} [1 - 2\mu < C_1 \Phi(x), \Phi(x_{obs}) >_{\mathcal{H}} + \mu^2] < C_1 \Phi(x), \Phi(x_{obs}) >_{\mathcal{H}}]^2] < 1 \\ \implies 0 < \mu < \frac{2}{< C_1 \Phi(x), \Phi(x_{obs}) >_{\mathcal{H}}} \end{aligned} \quad (22)$$

Hence, if μ is in the above range then the proposed algorithm converges. This bound (derived for non-linear systems) is similar to the general case of the bound of the convergence of step-size for Widrow-Hopf learning rule for a single node linear scenario.

VI. RESULTS

In this section, we present the simulation results based on the analysis presented in previous sections. An independently identically distributed (i.i.d) sequence $\{\pm 1\}$ was generated. Consequently, this sequence was passed through a non-linearity $f(x) = x - 0.9x^2$ as in [12] so as to simulate a non-linear system. Further, additive white Gaussian noise of variance 0.16 was added. In other words, we considered a simple de-noising problem for our simulations. The convergence and error performance of KLMS and diffusion-KLMS are shown in Fig. 1 for $A = C = [0.5 \ 0.5; 0.5 \ 0.5]$ and in Fig. 2 for $A = [0.666 \ 0.333; 0.333 \ 0.666], C = [0.5 \ 0.5; 0.5 \ 0.5]$. We see that although the LMS and diffusion LMS perform well in linear channels, they fail to converge in non-linear channels. We observe superior convergence to lower MSE floors is case of diffusion-KLMS as compared to KLMS, LMS, diffusion LMS and diffusion-RLS. We use $\mu = 0.2$ and spread parameter $\sigma = 0.1$ for KLMS and the proposed KLMS based approach. For LMS and diffusion-LMS, step-size $\mu = 0.02$ is used for simulation. We observe performance gain of two decades of the proposed approach with respect to LMS and diffusion-LMS. Also, we find a gain of a decade of performance with respect to single-node KLMS. We observe that the linear RLS exhibits poor performance in a non-linear scenario as the covariance matrix updation fails due to non-linearity.

In Fig. 3, the steady state behaviour of diffusion KLMS as a function of step-size where the theoretical curves, which are obtained from Section-V, are observed to be close to the experimental curves. The computational complexity of training phase of the proposed scheme is $O(D^2|\mathcal{G}|)$ and testing computational complexity is $O(D|\mathcal{G}|)$ as the computational complexity of the training and testing phases are given as $O(D^2)$ and $O(D)$ respectively as in [12] where D is the dimensionality of the observations.

From Fig. 4, we find that the proposed modeling of the transient behaviour of the MSE curve closely matches the experimental transient behaviour for diffusion-KLMS. Please note that the dynamical modeling for the algorithm is more accurate in the transient region of the plots. The transient region is generally specified by the time taken by the MSE plot to decay to $\exp(-1)$ of its initial value [21], which is also called time-constant of the adaptation. We see almost perfect modeling of MSE plot within the range of the time constant.

To study how MSE evolves as we remove or add another node in the network (or in another words change the network size), we plot the experimental MSE floor as a function of network size in Fig. 5. We see that as the network size increases the MSE floor decreases which is an intuitive result. Further, we compare the MSE floor obtained experimentally with the theoretical expression for the same A, C matrices for the given network size. We average over 1000 iterations with various choices of A and C , and plot their mean values both for theoretical and experimental MSE floors as a function of the network size. We see that the MSE floors as predicted by theoretical expressions derived in Section-V follow the experimentally obtained curves as we increase the size of the network.

VII. CONCLUSION

A new variant of KLMS algorithm has been proposed which is a distributed solution to the non-linear KLMS algorithm. The proposed algorithm converges to a lower MSE floor as compared to the original KLMS algorithm as shown in this paper. Theoretical expressions for both transient and steady-state performance have been derived which closely match with the experimental values. Hence, the proposed diffusion-KLMS is a better adaptive algorithm for estimation as compared to KLMS in distributed non-linear systems. This work has potential applications in non-linear distributed inference over some targeted 5G network's components like detection over massive MIMO and cooperative spectrum sensing for cognitive radio.

REFERENCES

- [1] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5G," *Communications Magazine, IEEE*, vol. 52, no. 2, pp. 74–80, 2014.
- [2] P. Li and R. C. De Lamare, "Adaptive decision-feedback detection with constellation constraints for MIMO systems," *Vehicular Technology, IEEE Transactions on*, vol. 61, no. 2, pp. 853–859, 2012.
- [3] C.-X. Wang, F. Haider, X. Gao, X.-H. You, Y. Yang, D. Yuan, H. Aggoune, H. Haas, S. Fletcher, and E. Hepsaydir, "Cellular architecture and key technologies for 5G wireless communication networks," *Communications Magazine, IEEE*, vol. 52, no. 2, pp. 122–130, 2014.
- [4] F. S. Cattivelli and A. H. Sayed, "Distributed detection over adaptive networks using diffusion adaptation," *Signal Processing, IEEE Transactions on*, vol. 59, no. 5, pp. 1917–1932, 2011.
- [5] E. Alpaydin, *Introduction to machine learning*. MIT press, 2004.
- [6] M. H. Hayes, *Statistical digital signal processing and modeling*. John Wiley & Sons, 2009.
- [7] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1035–1048, 2010.
- [8] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Transactions on Signal Processing*, vol. 55, no. 8, pp. 4064–4077, 2007.
- [9] X. Zhao and A. H. Sayed, "Performance limits for distributed estimation over LMS adaptive networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 10, pp. 5107–5124, 2012.
- [10] A. H. Sayed, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, 2014.
- [11] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1865–1877, 2008.
- [12] W. Liu, P. P. Pokharel, and J. C. Principe, "The kernel least-mean-square algorithm," *IEEE Transactions on Signal Processing*, vol. 56, no. 2, pp. 543–554, 2008.
- [13] N. Haghighat, H. Kalbkhani, M. G. Shayesteh, and M. Nouri, "Variable bit rate video traffic prediction based on kernel least mean square method," *IET Image Processing*, 2015.
- [14] B. Fan, W. Wu, K. Zheng, and W. Wang, "Proportional fair-based joint subcarrier and power allocation in relay-enhanced orthogonal frequency division multiplexing systems," *Communications, IET*, vol. 4, no. 10, pp. 1143–1152, 2010.
- [15] M. A. Z. Raja and N. I. Chaudhary, "Adaptive strategies for parameter estimation of box-jenkins systems," *IET Signal Processing*, vol. 8, no. 9, pp. 968–980, 2014.
- [16] K. Slavakis and S. Theodoridis, "Sliding window generalized kernel affine projection algorithm using projection mappings," *EURASIP Journal on Advances in Signal Processing*, vol. 2008, no. 1, p. 735351, 2008.
- [17] J. B. Predd, S. R. Kulkarni, and H. V. Poor, *Distributed learning in wireless sensor networks*. John Wiley & Sons: Chichester, UK, 2007.
- [18] P. Honeine, C. Richard, J. C. M. Bermudez, and H. Snoussi, "Distributed prediction of time series data with kernels and adaptive filtering techniques in sensor networks," in *42nd Asilomar Conference on Signals, Systems and Computers, 2008*. IEEE, 2008, pp. 246–250.
- [19] P. Honeine, C. Richard, J. C. M. Bermudez, H. Snoussi, M. Essoloh, and F. Vincent, "Functional estimation in hilbert space for distributed learning in wireless sensor networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing, 2009. ICASSP 2009*. IEEE, 2009, pp. 2861–2864.
- [20] A. Khalili, M. A. Tinati, A. Rastegarnia, and J. A. Chambers, "Steady-state analysis of diffusion LMS adaptive networks with noisy links," *IEEE Transactions on Signal Processing*, vol. 60, no. 2, pp. 974–979, 2012.
- [21] P. S. Diniz, *Adaptive filtering*. Springer, 1997.

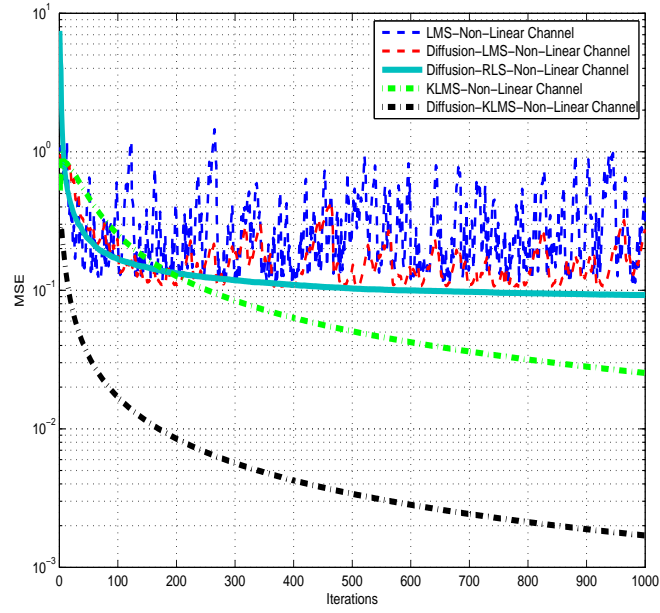


Fig. 1. Convergence plot for LMS, Diffusion-LMS, Diffusion-RLS, KLMS and Diffusion KLMS: $A = C = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$

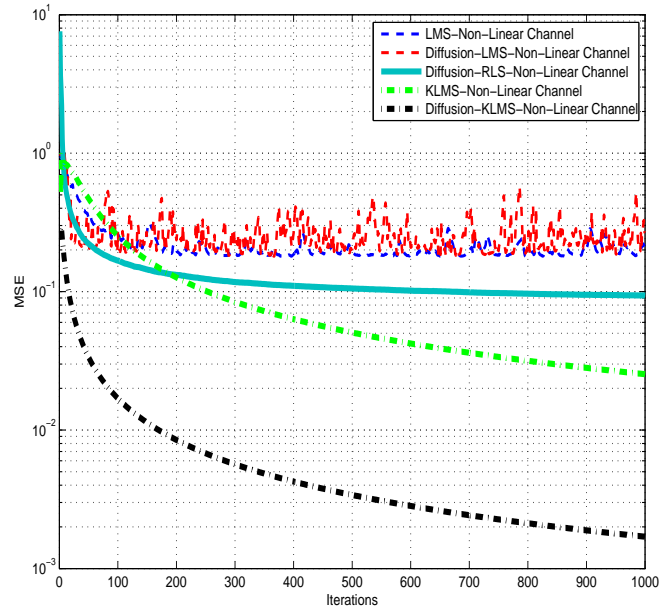


Fig. 2. Convergence plot for LMS, Diffusion-LMS, Diffusion-RLS, KLMS and Diffusion-KLMS comparison: $A = \begin{bmatrix} 0.666 & 0.333 \\ 0.333 & 0.666 \end{bmatrix}$, $C = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$

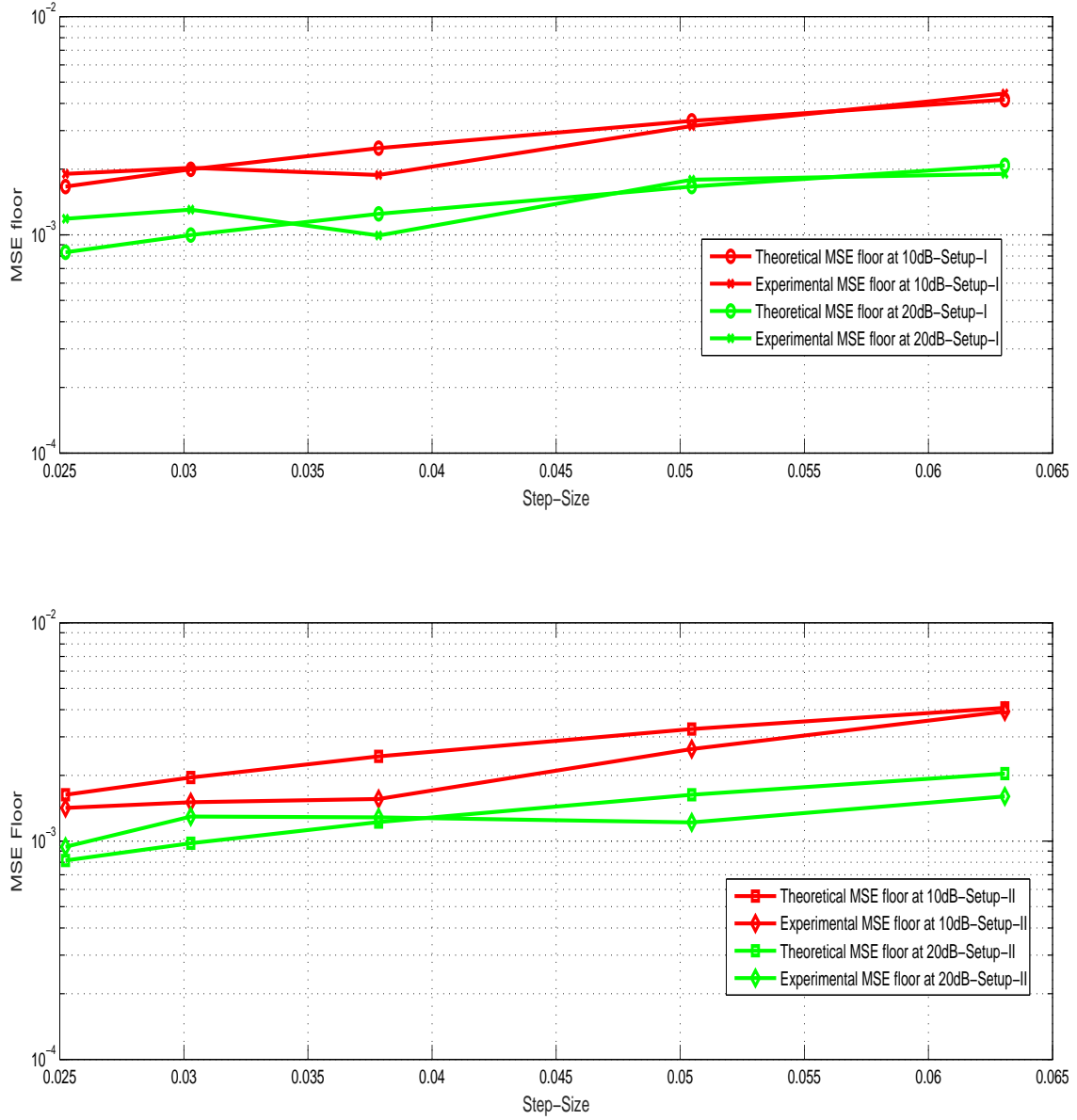


Fig. 3. MSE floors comparison for Diffusion-KLMS :Theoretical and Experimental; Setup-I: $A = [0.5 \ 0.5; 0.5 \ 0.5]$, $C = [0.5 \ 0.5; 0.5 \ 0.5]$, Setup-II: $A = [0.666 \ 0.333; 0.333 \ 0.666]$, $C = [0.5 \ 0.5; 0.5 \ 0.5]$

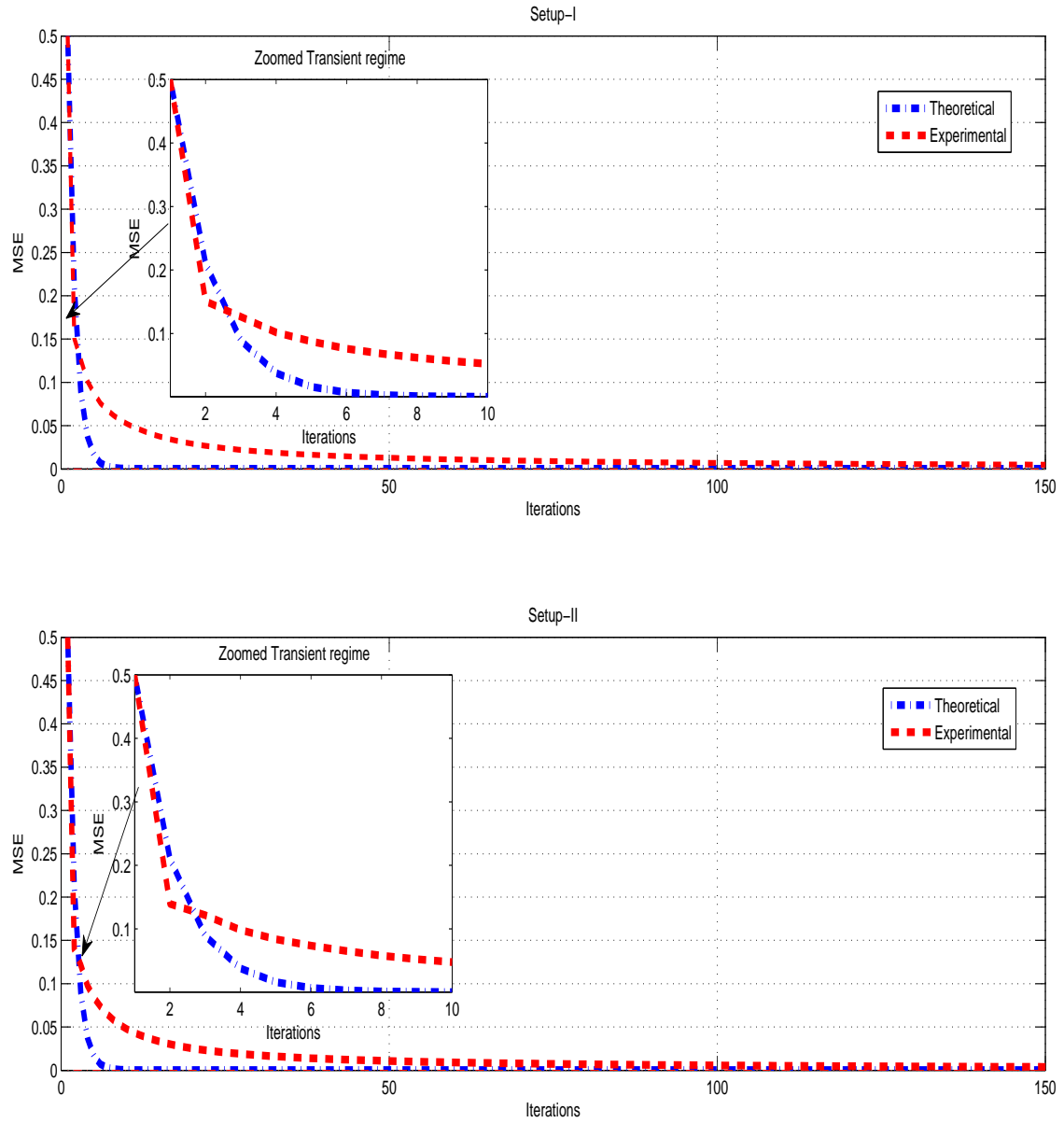


Fig. 4. Transient behaviour at step-size 0.12, Setup-I: $\mathbf{A} = \begin{bmatrix} 0.5 & 0.5; 0.5 & 0.5 \end{bmatrix}$, $\mathbf{C} = \begin{bmatrix} 0.5 & 0.5; 0.5 & 0.5 \end{bmatrix}$, Setup-II: $\mathbf{A} = \begin{bmatrix} 0.666 & 0.333; 0.333 & 0.666 \end{bmatrix}$, $\mathbf{C} = \begin{bmatrix} 0.5 & 0.5; 0.5 & 0.5 \end{bmatrix}$

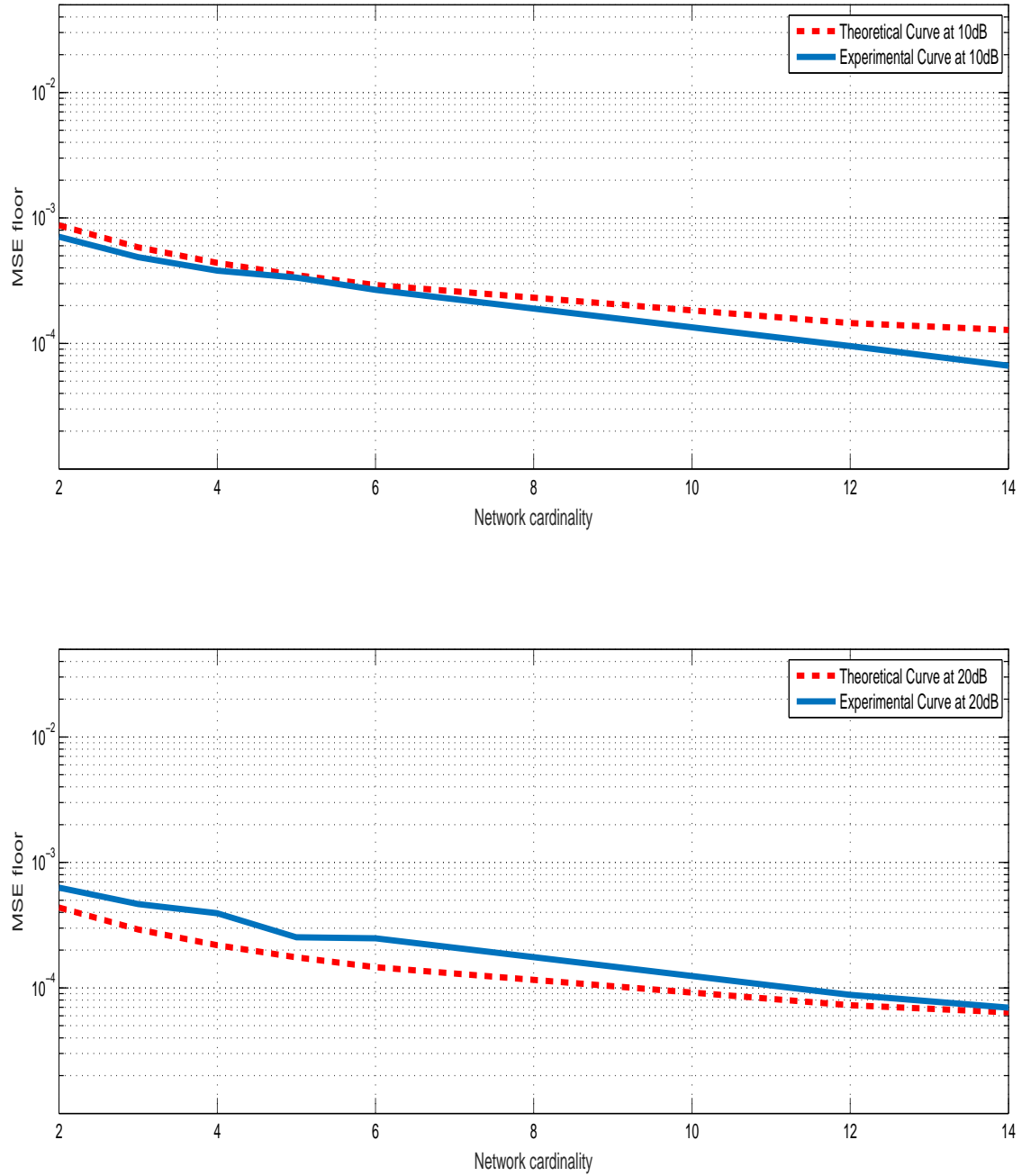


Fig. 5. Variation of MSE floor with number of nodes for SNR of 10dB and 20dB